GWD-R (Recommendations Track)
Grid Protocol Architecture Working Group

W. Johnston
Lawrence Berkeley National Lab and NASA Ames
J. Brooke
Manchester Computing, University of Manchester

http://grid.lbl.gov/GPA/
July 30, 2002

Core Grid Functions: A Minimal Architecture for Grids

Working Draft, Version 3.1

Status of this Memo: This memo is a draft for technical recommendations. Distribution is unlimited.

Abstract

This document is intended to define the ***current practice*** for a ***minimal*** set of Grid functions that provide uniform interfaces to architecturally, geographically, and administratively heterogeneous computing, data, and instrument systems that are managed by production Grids. By "production Grids" we mean the Grids that are trying to use Grid technology to provide services to a diverse user community to whom the operators of the Grid are responsible for providing a reliable and useful service. Defining these minimal services is very important because they represent the fundamental persistent infrastructure of the Grid. We use the term Core Grid Functions to represent this collection of Grid services that provide the persistent and most basic functionality of Grids.

Table of Contents

1. Introduction

This minimal set of Grid functions – the Core Grid Functions – are the services that provide the resource independence that will make the Grid a common infrastructure for all higher-level services. That is, they are the smallest set of services that are needed to build all other Grid frameworks, middleware, and applications. The minimal services may vary somewhat depending on the type of Grid resource – computing, data, instrument, etc. – and the set of minimal services for a given resource may not all run on the same platform: There may be functions for a resource that run on a "helper" system on behalf of the resource.

Defining a "minimal" set of functions is important because:

1) This set provides a metric about whether a system is a Grid enabled system, or not. Without the minimal set of functions, there will be Grid middleware, frameworks, and applications that cannot not function correctly, or at all, on such a system without adding the missing services.

2) Since the minimal set has to be installed and managed on every system that is a Grid resource they represent most of the operational effort in building and managing Grids.

3) These are ~~teh~~ the functions – together with operational policy – that enable the interoperation of different Grid implementations, different Virtual ~~Orgainzations~~Organizations, etc.

Our current understanding of Minimal Grid Functions includes

- persistent state, ~~and~~ registry, and ~~-~~ resource discovery
- resource scheduling
- uniform computing access
  - runtime / hosting environment for Unix style process interaction
  - runtime / hosting environment for OGSI style process interaction
  - means to establish application runtime environments
- uniform data access
- asynchronous information sources (events and monitoring)
- authentication, delegation, and secure communication
- authorization
- identity certificate management
- system management and access

All of these functions have elements that must be installed and managed on the Grid resources / systems, or have independent servers that provide the functions (e.g. discovery and identity certificates) and some may require both (e.g. asynchronous information sources require both services for generating events and a separate registry service).

The purpose of this document is primarily to identify functionality. The GGF topic-specific WGs are defining the protocols and APIs that implement these functions.

For each of the minimal set of functions, then we identify what are the minimal characteristics that must be standardized:

- functionality
- data structures (minimal?)
- bindings (is a C language binding minimal?)
- implementation (is a C language implementation minimal?)

The relationship of the Core Grid Functions to other elements of the Grid is indicated in Figure 1.

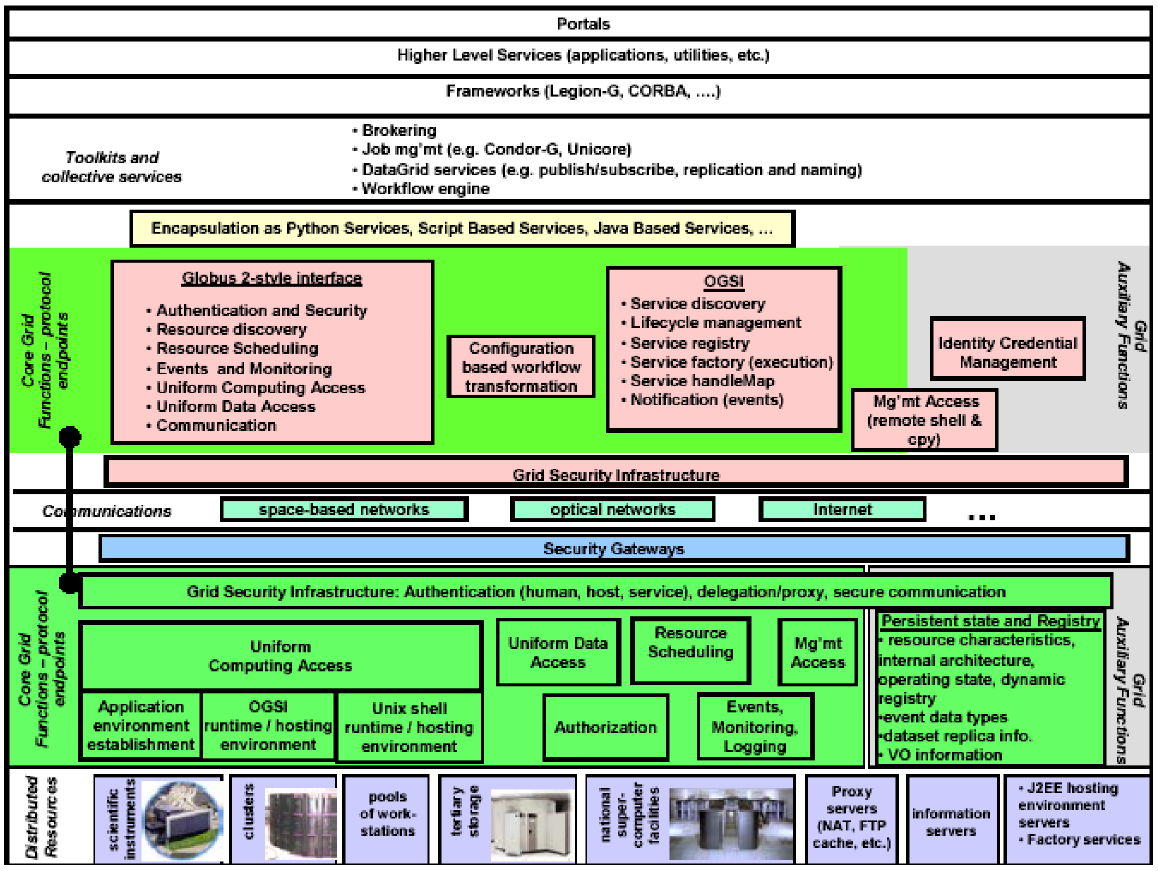
Portals
Higher Level Services (applications, utilities, etc.)
Frameworks (Legion-G, CORBA, ...)
Toolkits and collective services
• Brokering
• Job mg'mt (e.g. Condor-G, Unicore)
• DataGrid services (e.g. publish/subscribe, replication and naming)
• Workflow engine
Encapsulation as Python Services, Script Based Services, Java Based Services, ...
Core Grid Functions – protocol endpoints
Globus 2-style interface
• Authentication and Security
• Resource discovery
• Resource Scheduling
• Events and Monitoring
• Uniform Computing Access
• Uniform Data Access
• Communication
Configuration based workflow transformation
OGSI
• Service discovery
• Lifecycle management
• Service registry
• Service factory (execution)
• Service handleMap
• Notification (events)
Identity Credential Management
Mg'mt Access (remote shell & cpy)
Grid Auxiliary Functions
Grid Security Infrastructure
Communications
space-based networks
optical networks
Internet
...
Security Gateways
Grid Security Infrastructure: Authentication (human, host, service), delegation/proxy, secure communication
Core Grid Functions – protocol endpoints
Uniform Computing Access
Application environment establishment
OGSI runtime / hosting environment
Unix shell runtime / hosting environment
Uniform Data Access
Resource Scheduling
Mg'mt Access
Authorization
Events, Monitoring, Logging
Persistent state and Registry
• resource characteristics, internal architecture, operating state, dynamic registry
•event data types
•dataset replica info.
• VO information
Grid Auxiliary Functions
Distributed Resources
scientific instruments
clusters
pools of work-stations
tertiary storage
national super-computer facilities
Proxy servers (NAT, FTP cache, etc.)
information servers
• J2EE hosting environment servers
• Factory services

Figure 1. Core Grid Functions Relationships

2. Criteria for Minimality and Current Practice

In general, the criteria for a Core Grid Function is that it

- cannot be built on top of other Grid services,
- is essential for building other Grid services and applications, or for providing scalability or security,
- must be self contained (except possibly with respect to security).

~~Grid data and instrument resources may not have all of the functionality of computing resource, and may, therefore, not support a full set of CGF (e.g., Grid based scientific instruments and tertiary storage systems may not allow job initiation).~~

These criteria are evaluated for the candidate list of CGF below.

This document is also mostly intended to address "current practice." That is, there are groups that are building, or attempting to build, Grids and use these Grids to deliver services to well defined (though possibly diverse) user community. Most of these services exist today in one form or another. However, in a few cases the implementation work is turning up a few Grid functions that are missing, and we include these. When they are ~~and~~ functions that we understand well enough that we believe that they can be defined and specified relatively easily. The functions will be included here as well, since we believe that they are part of the minimal set.

> *Issue: Are there certain bindings that should be required on the client side as part of the minimal function definition?*

> *Issue: Should we identify minimal implementation characteristics of minimal functions – e.g. a C language implementation? (E.g., having only a Java implementation of a minimal function will exclude lots of potential Grid resources.)*

3. Core Grid Functions

3.1 Persistent State and Registry - Resource Discovery

A Grid information service must provide information about all Grid resources (including services), and should minimize the number of persistent information servers that have to be managed in order to enable Grid services and applications. This function provides the basic persistent state and registry mechanism for the Grid.

Functionality

The persistent state and registry function should

- Provide for locating all Grid resources with specified properties, within a certain scoping, and without relying on user maintained enumerated lists
- Provide state information (directly or as pointers to other services)
- Accommodate a dynamic resource base
- Accommodate data from users, Virtual Organizations, applications
- Be extensible to "all" Grid persistent state – that is, all Grid services can be sources of information, and if this information needs to be referenced and/or discovered, it should be possible to store and/or represent it in the Grid information service. E.g.
  - Data from users, Virtual Organizations, applications
  - Computing resource characteristics~~s~~
  - Available software
  - Current user allocation
  - Asynchronous Information Sources registry and data content
- ~~Accommodate data from users, Virtual Organizations, applications~~

  ~~Provide all of this within~~That is, there should be a unified information representation and discovery framework so that the number and variety of persistent servers that have to be supported is minimized

Essential characteristics

- query
- query access control
- soft state data entry
- data entry access control
- search scoping
  - hierarchical
  - "views" (e.g. MDS-2 index server)`
- aggregated queries
  - query all of the information sources within the scope of a information service

Issues

- Are aggregated queries – query all of the information sources within the scope of a information service – essential for scalability?

QoS

- query response time

- data entry limitations

Support Required on Grid Resource Platforms

- Soft state registration mechanism

Environmental Support Required

- Typically one or more servers must be managed at a site and/or in a VO

Is this a minimal service?

Yes, minimal:

- Discovery is an essential Grid function. Without discovery, you cannot build virtual systems from dynamically changing pools of resources.
- Management of persistent servers is operationally expensive, therefore for VOs and Grids to be operationally successful, it is critical to minimize the number of servers needed for a persistent Grid.
  - Storing / representing all manner of persistent Grid information with one service is important to minimize required operational support.

No, not minimal:

- no issues

Cannot address at this time:

- no issues

Current Experience

- Globus MDS-2 [1]
- Current GGF docs

3.2 Resource Scheduling

Scheduling is separate from process initiation. It may involve many different types of resources, some of which do not involve processes.

Functionality

- Establish a given, on-demand, virtual system relationship among an administratively independent set of Grid resources – that is, among all of the components that need to interoperate (e.g. computing resources for parallel, pipelined, multi-level processes) to accomplish a task in the Grid environment that involves many coordinated elements
- Return information sufficient for negotiation of a common QoS (e.g. time slot) among independent resources

Characteristics

- Co-scheduling by negotiation
- Time-of-day reservation
  - Hard (e.g. at a fixed time)
  - Soft (e.g. at some time within a specified interval)
- Accounting
- Security
  - Access control based on the Grid identity / DN (distinguished name – see section 3.6. “Remote Authentication, Delegation, and Secure Communication)

Issues

- How is a reservation guaranteed
- How to control who gets to reserve a resource

QoS

- ~~This~~ Resource scheduling is a key function for implementing QoS

Support Required on Grid Resource Platforms

A scheduler operating on the resource must

- Provide time of day reservation
- Evaluate the future availability of a reservation request and pass that information back to the requester
- Support soft reservations to allow time for an external broker to negotiate a common reservation among several resources

Environmental Support Required

- none

Is this a minimal service?

Yes, minimal:
  - Essential for QoS
  - Not possible to emulate

No, not minimal:

- no issues

Cannot address at this time:

- no issues

Current Experience

- GGF
  - SchedWD8 - "Super Scheduler Steps/Framework", J. Schopf, 7/01 - overview of current user practices for scheduling across administrative domains. [2]
- Globus
  - SNAP/GARA –" SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems" - [3]
- Resource schedulers
  - PBSPro [4]
  - Maui Silver [5]

3.3 Uniform Computing Access

Issue: There is a general problem in this section about the distinction and scope of "hosting," "runtime," and "execution" environments.

wej: Propose the following definitions:

1) "execution environment" is what is needed by the application in order to run

2) "hosting environment" is the standard runtime systems provided by
a) the Unix shell
b) the OGSA hosting environment

3) "resource" environment = hosting environment + platform policy/configuration

The text below has mostly not been modified to reflect this, except where I think that the use of the term "hosting" conflicts with the OGSA usage.

Most current Grid access to computing resources involves sending a script to the remote systems and then either exec-ing a shell and passing the script to the shell, or, more commonly for a purely computing resource, submitting the script to a batch queuing system.

To be truly described as uniform computing access there has to be a means for composing a common structure for a request for computing resources that is independent of different policies / configurations and naming conventions at individual sites or machines. A request for a computing resource also implicitly involves (for example) a request for staging areas for files, areas for storage of results, temporary workspace while the request is running. A problem is that there is no agreed convention for providing this holding environment for the computing request and unless the request for resources is conformant with the local site policy[1] it will fail. To take just one example, supposing a job is submitted under a temporary account validated by some certificate mapping policy. If the site has a policy that all filestore is deleted at the end of the running of the job, then a request that assumes that the files are held to be post-processed at a later stage will lose all such results. This also applies to an implicit assumption that pre-request staging is available. This becomes particularly important when we are chaining jobs or running workflow requests for computation and data access across several resource centers with dependencies between the different stages of the workflow. Clearly without pre and post compute resource file-staging such complex workflows cannot be guaranteed over multiple sites since we cannot usually guarantee that the next link in the job chain will be available before the last link ends its computation so that the results can be transferred to the next stage.

A request for a resource on the Grid therefore needs a working environment that is sufficiently robust to allow multi-site workflows to be constructed. These working environments need to be described in sufficiently abstract terms so that they can apply across a sufficiently wide range of policies. In other words the application execution ~~hosting~~ environment for ~~the~~ a Grid job ~~resource~~ needs to be described so that it can be implemented in multiple ways, on particular resources, implementing particular policies.

Even on a single resource, ~~the appropriate~~ it must be possible to adapt to local policies (that may have nothing to do with the runtime / hosting environments) ~~must be in place~~ for jobs to run.

It is important to note that not all workflow requests on the Grid are submitted to computers. A~~, a~~ Grid workflow may include experimental or observational apparatus, e.g. telescopes,

[1] The term "policy" is used here in a general sence that includes site configuration, administrative based usage requirements, user authorization, etc. All of these must be taken into account in order to successfully run a job.

microscopes, robotically controlled laboratory experiments. It is thus a key Grid issue that facilities for data staging, storage, and transmission are supplied~~-~~ in some fashion in order to accommodate the application.~~as needed in a hosting environment~~. Such apparatus may not have such functionality supplied in its pre-Grid existence, since it may not be needed in stand-alone mode. Thus to provide minimal functionality it may be necessary to attach a server to the apparatus to provide the Grid hosting environment. The expression of Grid workflow requests via well-constructed abstractions can help to give guidance as to what functions such a server should support.

3.3.1 Process Initiation and Policy / Platform Configuration Accommodation

Functionality

- Initiate a process or task script on a the ~~remote~~ Grid resource ~~system~~
- Support queries about queue types (different classes of service)
- Support submission to named~~s~~ queues (different classes of service)
- Perform access control based on Grid identity / DN
- Support pre and post job staging areas for data holding and storage.
- Hosting environments, possibly on separate platforms that are capable of supporting such environments, ~~especially~~ for facilities attached to the Grid~~ which are not computers~~.

Characteristics

Based on the abstractions of a workflow composed of atomic requests for computational, storage, networking and other resources (e.g. control of an experimental apparatus), a minimal Grid service for common access will provide mechanisms for mapping the abstractions of the work request and its ~~hosting~~ execution environment into individual site specific policies.

> ***Unicore Example:***~~-~~*One model for this is provided by the Unicore project [6] where the request or workflow of requests is defined as a Abstract Job Object where all the details of the workflow and its environment are defined as a hierarchy of Java classes. A requesting client can create an instantiation of the AJO which is sent across the Internet to sites that are requested to implement all or part of it, and to transfer the next part of the workflow to another site. At each site the relevant portion of the AJO is "incarnated" or translated into the site specific scripts necessary for it to actually run on the sites resources under the sites' policies. The AJO concept is recursive, in that AJOs can contain sub-AJOs to build a model of a workflow as a Directed Acyclic Graph.*
>
> *A key feature of this approach is that the job preparation client is restricted in the way the multi-site job is prepared, restricted by the structure inherent in the AJO. In this way certain generic assumptions about user space, file transfer, and subjob dependency are supported as the job is composed. Another way of looking at this is to say that the abstractions of the hosting environment are implicit in the creation of the AJO. This is restrictive, since nothing can be done outside of the permitted abstractions, thus one cannot make use of specific knowledge of what might happen if the job were to be run at a particular site. Abstractions for compiler type, software resource availability, license availability can and are being built into the AJO concept which is extensible via the object-oriented model (e.g. via inheritance from more general job object classes). This model requires that each resource on the Grid has a mechanism for receiving such an AJO and translating (or "incarnating") it into the specific commands and scripts necessary to run the AJO or sub-component of the AJO at a particular site. In Unicore this is done via an Incarnation Database, but this need not be part of a minimal Grid service since it could be built via interrogation of Grid Information Systems such as the MDS. This requires that information on site policy is available via GIS.*

However the key feature of a Uniform Computing Access service must be the ability to describe all features of the resource request (or job) including the essential features of the ~~hosting~~ required execution environment in a manner which is independent of policies at individual sites. If this is not met then the individual policy of the site at which the component of the job request will be run must be determined as the job is being composed, this is not scalable and not resilient. In particular it makes migration of a job component to another site problematic since the policy differences may cause the component to fail unless renegotiation and alteration of the job scripts is carried out before migration. This would seem highly impractical. These problems are all compounded for multi-component workflows with dependencies between different components. The way the site policy impacts on the job component may well be dependent on the identity of the job owner. Authentication is via certificate (see the Grid Services defined in Section 3.6) but from this there must be access to information on the authorization rights of such a user at a given site. This, however, fits with the approach described in the example above ~~proposed here~~ since the Incarnation Data Base can hold such information, or pointers to where such information resides at the site. This concept has also been expressed as a "Policy Engine" [7] [8].

Issues

- There are both practical and actual issues that arise with this service that may make different Grid computing resources appear to have a uniform interface, but they will actually be heterogeneous in that a task sequence that works on one system will not work on another due to the system configuration/architecture.
  - Most of these configuration issues relate to either setting up the environment needed to execute the Grid job, or in pre- and post staging data.
  - The issues with staging data derive from the fact that cluster-like systems (which is to say, most of today's large computing resources) may
    - not provide all services on every node. For example,
      - some nodes may not have host certificates, and therefore services requiring host certs (like GridFTP server) will not work.
      - some nodes may not have access to the Internet
    - ~~–~~ ~~Have~~ have different approaches to how the file system environment for a job/task is established
    - ~~–~~ ~~Have~~ have different ways to stage and store files locally that must be accessed by computing nodes of the resource
- Some task models require pre-staging data independently of the management of the computation – the Abstract Job Object framework of Unicore provides a possible solution, and it is important to see how this might work with other resource management middleware.
- Protocols and standards for providing a common description of Grid resources and ~~hosting~~ policy environment characteristics~~s~~ need to be agreed. This must provide information about the computing environment architecture and other details necessary for successful running of the job.
  - The local sites can then advertise their resources and policies in sufficient detail that Grid middleware can translate from the protocols for common access to the site-specific scripts which will "incarnate" or "ground" the work request.
- Support for reception and onward transmission of multi-site workflows needs to be included in the hosting environment.
- Terminology: "hosting" ~~or~~ vs. "runtime" vs. "execution" environments?

QoS

- ~~a~~ A policy engine can prevent job failure due to ~~hosting~~ runtime / execution environment mismatches, administrative issues, etc., by evaluating the suitability of an environment prior to committing a job to that environment

- Co-scheduling, etc., is provided by the scheduling service

Support Required on Grid Resource Platforms

- Transfer the Grid identity / DN of users that initiate jobs into the accounting records for those jobs
- A~~Provide a~~n access control mechanism based on the Grid identity / DN must be provided and managed
- A Grid process initiation service must be installed and managed
- ~~Access control mechanisms must be maintained~~
- Staging areas for workflow support need to be provided as part of a Grid hosting environment.

Environmental Support Required

- Need to provide information for translating from common resource description protocols to local definitions. Resource and policy databases are one means to achieve this.

Is this a minimal service?

Yes, minimal:

An essential Grid service.

No, not minimal:

no issues

Cannot address at this time:

no issues

Current Experience

- Globus gatekeeper, GRAM [9]
- Unicore Project [6]
- EZGrid [8], [7]
- Current GGF docs

3.3.2 Unix Hosting / Runtime Environment

Functionality

-

Characteristics

-

Issues

-

QoS

-

Support Required on Grid Resource Platforms

-

Environmental Support Required

-

Is this a minimal service?

Yes, minimal:

An essential service.

No, not minimal:

no issues

Cannot address at this time:

no issues

Current Experience

-

3.3.3 OGSA Hosting Environment

Functionality

-

Characteristics

-

Issues

-

QoS

-

Support Required on Grid Resource Platforms

-

Environmental Support Required

-

Is this a minimal service?

Yes, minimal:

An essential service.

No, not minimal:

no issues

Cannot address at this time:

no issues

Current Experience

-

3.4 Uniform Data Access

Today the primary Grid data access service is access to named, unstructured objects (e.g. "flat" files). That is, objects / files whose structure is understood only by the application that reads the files, and not by the storage system. Hence, the primary current model for Grid data access is FTP.

Other functionality is emerging in Grid storage resources:

- Support for some mechanism of sub-setting or filtering data before it leaves the storage resource.
- providing access to relational databases.

Issue: How many core access types are there?
Current experience is with objects / files, relational DBMS, and (maybe) Object Oriented DBMS.

3.4.1 Flat File / Unstructured Object Access

Functionality

- Storage access abstraction
- Partial file access
- Integrated Grid security infrastructure security and access control based on the Grid identity / DN

Characteristics

- Separate control and data channels
  - So that control channels may be authenticated and encrypted while data channels may be as efficient as possible
- Third-party transfers (e.g., between GridFTP servers)
- Wide area network communication parameter optimization
- Integrated performance monitoring instrumentation
- Network parallel transfer streams
- Support for proxies (NAT, cache)
- Server side data striping (e.g. DPSS [10] and HPSS striped tapes)
- Server-side computation / filtering
- Support for tertiary storage system access
  - Batching of file requests
  - Tape / other near-line media pre-stage requests
  - Pinning files in the storage system cache
  - Other tape / other near-line media issues?

Issues

- Most of the characteristics above are essential for scalability / performance rather than functionality
- Support for tertiary storage system access (as above)
  - if not in GridFTP (e.g.), then where?
  - tertiary storage cannot function correctly / optimally without this sort of functionality

- Is reliability/restart for large file transfers a minimal characteristic ? Can it be built on top of the other features? (Probably, given partial file reads.)
- We have little experience with server-side computation
- Naming abstraction is important, but can be provided by higher level services (e.g. metadata catalogues and naming services)

QoS

- Scheduling datasets to be immediately available (e.g. pre staging of tapes, pinning files in caches) is probably required for QoS

Support Required on Grid Resource Platforms

- Data access server
- Grid security infrastructure (credentials, protocol libraries)
- Grid identity / DN based access control

Environmental Support Required

- none

Is this a minimal service?

Yes, minimal:

An essential service.

No, not minimal:

no issues

Cannot address at this time:

no issues

Current Experience

- GridFTP [11]
- SRB/MCAT [12]

3.4.2 ~~Relational Data Base access~~Relational Data Base Access

Functionality

-

Characteristics

-

Issues

-

QoS

-

Support Required on Grid Resource Platforms

-

Environmental Support Required

-

Is this a minimal services issue?

Yes, minimal:

- If there is a RDMS server on a Grid resource, then at least a Grid front end is probably needed for Grid authentication and access control (?)

No, not minimal:

- Can this be built on top of another Grid service? (Secure remote shell?)

Cannot address at this time:

- Do universal standard access methods currently exist?

Current Experience

- SRB/MCAT, RDBMS access [ref]
- GGF, Database Access and Integration Services Working Group
  - *Grid Data Services - Relational Database Management Systems (Version 1)*. This paper discusses issues associated with the development of relational database services, including usage scenarios. [13]
- EU DataGrid
  - *Project Spitfire - Towards Grid Web Service Databases* . This paper describes the Spitfire Grid database access service for relational databases. [14]

3.4.3 Object Oriented Data Base access

Functionality

-

Characteristics

-

Issues

-

QoS

-

Support Required on Grid Resource Platforms

-

Environmental Support Required

-

Is this a minimal services issue?

Yes, minimal:

?

No, not minimal:

?

Cannot address at this time:

- No universal standard access methods currently exist (?)

Current Experience

- *Pursuit of a Scalable High Performance Multi-Petabyte Database* [15]
- *Creating Large Scale Database Servers* [16]
- *Objectivity Open File System* [17]

3.5 Asynchronous Information Sources (Events, Monitoring, Logging, etc.)

We use the term "Asynchronous Information Sources" (AIS) to mean any source of XML formatted objects that can publish its existence and object content characteristics, and then support subscription based delivery of those objects.

A model for this service is provided by the Grid Monitoring Architecture [18]. This model involves sources/producers that publish their existence and the characteristics of the data content that they supply by sending this information to a registry service. The data sinks/consumers search the registries for the desired data characteristics, and then subscribe directly with the source/producer for data delivery.

Asynchronous Information Sources include things like

- Events (system, application, workflow scripts)
- Monitoring (system parameters, batch schedulers, network, application (e.g., see [19]))
- Accounting / auditing records
- Soft state registration (of anything that changes)
- Logging of all sorts

Soft state registration, and other targeted uses (e.g. accounting, logging), should be able to be accomplished with this mechanism by, e.g.,

1. having the application implement an AIS sink/consumer function;
2. having the application implement an AIS registry function;
3. having the targeted-use source specifically register with the application sink (e.g. a logging or accounting application);
4. then have the sink automatically subscribe to everything that gets registered in its registry – i.e. all of the data sources needed for the application

Functionality

- Source registration (a la GMA, the source registers its existence and the content of the objects that it will generate)
- Registry should be "globally" searchable based on various source and/or AIS object content characteristics
- Receiving data is by subscription and by direct transfer (source to sink) – the GMA model

Characteristics

- AIS sources should be able to register with whichever registries are appropriate, of which there may be many
- Data delivery (source to sink) should be available with multiple transport semantics
  - Streams
  - Messages
  - Unreliable multicast
  - Reliable multicast
  - Transactional
- Registry search semantics should be relational among the named AIS object fields

Issues

- Should a sink/consumer be able to "trigger" a refresh of a source/producer monitor? How?

QoS

- No QoS issues (?)

Support Required on Grid Resource Platforms

Basic monitoring functions with GMA source semantics need to be installed and managed.

Environmental Support Required

Registry servers must be provided and maintained.

Is this a minimal service?

Yes, minimal:

- Cannot, in general, be built using the job initiation service
  - Users cannot, in general, start and maintain long lived servers on Grid resources
  - There will be Grid system that we need to monitor, but which will not support a job initiation service (e.g. instruments and storage systems)

No, not minimal:

no issues

Cannot address at this time:

no issues

Current Experience

- GGF GMA docs
- Implementations
  - *Distributed Monitoring Framework (DMF),* [20]
  - *Information and Monitoring Services Architecture*, [21]
  - *A Framework for Control and Observation in Distributed Environments*, [22]
- Examples
  - *Monitoring Data Archives for Grid Environments*, [19]

3.6 Remote Authentication, Delegation, and Secure Communication

Remote authentication is accomplished by techniques that verify a cryptographic identity in a way that establishes trust[2] in an unbroken chain from the relying party back to a named human, system, or service identity. This is accomplished in a sequence of trusted steps, each one of which is essential in order to get from accepting a remote user on a Grid resource back to a named entity.

Delegation involves generating and sending a proxy certificate and its private key to a remote Grid system so that remote system may act on behalf of the user. This is the essence of the single sing-on provided by the Grid: A user / entity proves its identity once, and then delegates its authority to remote systems for subsequent processing steps.

A secure communication channel is derived from the Public Key Infrastructure process and the IETF Transport Level Security protocol, as described below.

The trust establishment process involves:

1. Binding an entity identity to a Distinguished Name (“DN” - the subject name in an X.509 identity certificate)
   - Trust in this step is accomplished through the (published and audited) policy based identity verification procedures of the Certification Authority that issues the identity certificates
2. Binding a public key to the DN (generating an X.509 certificate)
   - Trust in this step is accomplished through the (published and audited) policy based operational procedures of the issuing Certification Authority (“CA”).
3. Assurance that the public key that is presented actually represents the user
   - Trust in this step comes from the cryptography and protocols of Public Key Infrastructure.
4. Assurance that a message tied to the entity DN could only have originated with that entity:
   - Trust that a message signed by a private key could only have been signed by the private key corresponding to the public key (and therefore the named entity via X.509 certs) comes from public key cryptography
   - Trust in this step is also through user key management (the mechanism by which the user limits the use of its identity), which is assured by user education, care in dealing with one’s cyber environment, and shared understanding as to the significance of the private key.
5. Mutual authentication, whereby two ends of a communication channel agree on each other’s identity
   - Trust in this step is through the cryptographic techniques and protocols of the Transport Level Security (“TLS”) standard.
6. Delegation of identity to remote Grid systems
   - Trust in this step is through the cryptographic techniques and protocols for generating, managing, and using proxy certificates that are directly derived from the CA issued identity certificates.

At this point a cryptographic proxy identity is present at the remote Grid resources, and it is trusted to represent the original named entity.

By virtue of using TLS for mutual authentication, a secure communication channel has been established at this point. The channel uses symmetric key cryptography, and session key management for this secure channel is part of the TLS protocol.

[2] Trust is “confidence in or reliance on some quality or attribute of a person or thing, or the truth of a statement.” Oxford English Dictionary, Second Edition (1989). Oxford University Press.

Confidentiality is a characteristic of the TLS channel.

Message integrity comes with the TLS channel, but not without encryption. The GSS-API provides an integrity-only function.

3.6.1 Certification Authority and Certificate Management

The CA accomplishes steps 1 and 2 in the trust establishment process.

CAs must have clear and published policy on the circumstances under which they will issue identity certificates, and how the DNs are generated. While this policy is not uniform, it is public so that each virtual or actual organization that supplies resources to the Grid may make an informed decision on whether to accept the remote user identity certificates or not.

CAs have clear and published policy on ~~there~~ their operating procedures that indicate the level of care taken in the certificate generating and issuing process in order to ensure that certificates traceable back to the CA are not forged or otherwise cryptographically compromised.

Functionality

- Provides a mechanism for users / entities to request certificates
- Provides a registration process that verifies user/entity identity
- Issues and signs X.509 identity certificates
- Provides Certificate Revocation List generation, management, access, and use
- Provides a certificate repository
- Has a formal policy

Characteristics

- Operated according to a formal, published policy statement, and with some level of audit
- Highly secure
- Formal logging

Issues

- Negotiating common policy among multi-institutional and/or international VOs is hard
- Is a certificate repository a minimal function?
- Certificate Revocation List distribution protocols and management are not well defined / widely used yet (?)

QoS

- No QoS issues (?)

Support Required on Grid Resource Platforms

- The CA public key must be installed on all platforms that authenticate Grid entities via the Grid security infrastructure – for a mixed user population this will involve multiple CAs
- The CA signing policy file must be installed on all platforms that authenticate Grid entities via the Grid security infrastructure (this allows a relying party to restrict the certs that it will accept from a given CA based on the name space of the DN)

Environmental Support Required

- A secure physical and cyber infrastructure are needed for the CA

Is this a minimal service?

Yes, minimal:

- The Certification Authority, and its published policies, are an essential component for this service for establishing trust in remote user access

No, not minimal:

no issues

Cannot address at this time:

no issues

Current Experience

- ~~See~~ T~~t~~he DOE Science Grid CA (doegrids.org, [23]) ~~for~~ provides an example of a production, multiple VO CA
  - See the project site for the DOE Science Grid CA (http://envisage.es.net) which has the project documents and history [24]
- The EU DataGrid provides several examples of production CAs and policy
  - Certification Authorities – see [25]
  - Cross certification check list – see [26]

3.6.2 User Key Management

The user / entity private key corresponding to the public key that the CA has bound to the DN is what the remote entity uses to prove that it is the entity represented in the certificate. This key represents the user identity and must be carefully protected. The CA must convey the importance of protecting the private key to the user when the certificate is issued.

At least one cryptographic mechanism must be provided for protecting the private key. Typical mechanisms are to keep the private key encrypted with a second (symmetric) key that is protected either with a passphrase, or kept ~~keep it~~ in a cryptographic device such as a smart card.

3.6.3 Mutual Authentication

The Transport Level Security protocol (formerly know as SSL) uses a cryptographic protocol, together the tracability of the certificates back to an issuing authority that is policy based, to establish an authenticated and secure communication channel.

Functionality

- Provides for using host identity credentials at both ends of a transport connection for
  - validating the system identities
  - securely conveying user / Grid entity credentials / proxy to the remote system

Characteristics

- Provided by IETF TLS

Issues

- none

QoS

- No QoS issues (?)

Support Required on Grid Resource Platforms

- Host identity certificates must be installed on the Grid resource systems
- Host certificates must be issued by a CA according to a clear policy
- Compatible cryptographic libraries must be installed on the receivers (e.g. OpenSSL)

Environmental Support Required

- Care must be taken to keep the TLS implementation up-to-date. Several vulnerabilities have been detected and corrected in the recent past.

Is this a minimal service?

Yes, minimal:

This is an essential component for the Grid security service to securely access a remote system

No, not minimal:

no issues

Cannot address at this time:

no issues

Current Experience

- Globus GSI [27]
- OpenSSL [28]

3.6.4 Secure Communication

Following mutual authentication, the Transport Level Security protocol (formerly know as SSL) uses a cryptographic protocol to establish a secure communication channel.

Functionality

- A secure, stream oriented communication

Characteristics

- Provided by IETF TLS

Issues

- As above (section 3.6.3, “Mutual Authentication”)

QoS

- As above (section 3.6.3, “Mutual Authentication”)

Support Required on Grid Resource Platforms

- As above (section 3.6.3, “Mutual Authentication”)

Environmental Support Required

- As above (section 3.6.3, “Mutual Authentication”)

Is this a minimal service?

Yes, minimal:

As above (section 3.6.3, “Mutual Authentication”)

No, not minimal:

no issues

Cannot address at this time:

no issues

Current Experience

- As above (section 3.6.3, "Mutual Authentication")

3.6.5 Delegation

Delegation is the process by which a user's identity is carried to a remote system without the user being directly involved at the remote system.

This involves generating a proxy certificate that is derived from the user's identity certificate. The proxy, its private key, and the user identity certificate are all conveyed to the remote systems in order to support authentication and authorization, and to support conveying the user identity to subsequent systems that may be needed for the Grid task.

This is accomplished with a delegation protocol.

Functionality

- Generates "proxy" certificates
- Provides a protocol for conveying the proxy to the remote site

Characteristics

Issues

- Clearly defined API for delegation
- Limited delegation

QoS

- No QoS issues (?)

Support Required on Grid Resource Platforms

- Software support for the delegation process

Environmental Support Required

- The private key of the proxy is stored in a file that should be "well protected." E.g., it should only be readable by the UID of the process that must use that key to generate downstream proxies.

Is this a minimal service?

Yes, minimal:

Delegation is an essential Grid service. For a complex Grid task operating in a large Grid, it would be virtually impossible for the user to directly interact with every system that might be involved.

No, not minimal:

no issues

Cannot address at this time:

no issues

Current Experience

- Globus GSI [27]
- Draft TLS extensions for delegation [29]
- GSS-API extensions for Grids [30]
- GSI Online Credential Retrieval – Requirements [31]

3.6.6 GSS-API

The IETF GSS-API provides an API for security context establishment, message integrity, and message confidentiality.

Functionality

- An API for security context establishment, message integrity, and message confidentiality

Characteristics

Issues

- GSS-API implementations also do framing, which makes GSS-API more then just an API. This means that the receiver must understand the GSS framing.

QoS

- No QoS issues (?)

Support Required on Grid Resource Platforms

- GSS libraries

Environmental Support Required

- none

Is this a minimal service?

Yes, minimal:
- No other GSI service provides message integrity without encryption

No, not minimal:
- Is message integrity w/o encryption essential?

Cannot address at this time:

Current Experience

- Globus GSI [27]
- GSS API [32]

3.6.7 The Overall Grid Security Infrastructure Service

Functionality

- As above

Characteristics

Issues

- Well defined APIs for the basic functions

- GSS-API issue as above
- Interfaces to local security domains that are not PKI, e.g. Kerberos
- On-line credential repositories
- CRLs
- Is basic ACL (access control list – e.g. the Globus mapfile) authorization a minimal function?

QoS

- No QoS issues (?)

Support Required on Grid Resource Platforms

- See individual components
- ACLs for authorization

Environmental Support Required

- See individual components

Is this a minimal service?

Yes, minimal:

- Secure authentication and communication are essential Grid functions
- Trust by the relying party (the remote Grid resource) in the identity of the entity seeking to establish a secure communication channel is an essential service. This service actually has several different components, all of which are required to provide the service.
- Secure, authenticated transport is essential for Grid service command channels / messages

No, not minimal:

- Authorization is a site policy issue

Cannot address at this time:

?

Current Experience

- PKI [33]
- Globus implementation of the Grid Security Infrastructure [27]
- Globus implementation of GSS-API [27]

3.6.8 Credential Repositories

Proxy credential repositories and user credential repositories may be essential for the usability of the Grid.

Functionality

-

Characteristics

-

Issues

-

QoS

-

Support Required on Grid Resource Platforms

-

Environmental Support Required

-

Is this a minimal services issue?

Yes, minimal:

?

No, not minimal:

?

Cannot address at this time:

-

Current Experience

- *myProxy server {NCSA, 2002 #161} {Novotny, 2001 #160}*
-

3.7 System Management and Access

System management, and sometimes remote user access, are needed so that Grid resources may be managed and interactively accessed within the Grid context.

Functionality

- Remote login, authenticated and secured with Grid security functions and authorization based on a Grid identity / DN
- Remote shell, authenticated and secured with Grid security functions and authorization based on a Grid identity / DN
- Remote copy, authenticated and secured with Grid security functions and authorization based on a Grid identity / DN

Characteristics

- Secure
- Capable of managing / forwarding the Unix hosting/runtime functions of standard in/out/error

Issues

-

QoS

- No QoS issues (?)

Support Required on Grid Resource Platforms

- The servers that support these services must be installed and maintained.
- Host certificates must be installed and managed
- User CA keys must be installed and managed
- ACLs for authorization must be installed and managed

Environmental Support Required

- none

Is this a minimal service?

Yes, minimal:

– This seems to be an essential service, because if it is not provided within the Grid context then it is always accomplished in a ad-hoc manner.

No, not minimal:

?

Cannot address at this time:

no issues

Current Experience

- GSIssh [34]

3.8 Architectural Constraints

In order to be called a Grid Common Service, it should not be possible to convey command and control messages to remote Grid systems except through the secure and authenticated communication provided by the Grid security functions. This is indicated pictorially in Figure 1, “Core Grid Functions Relationships.”

A Grid without this sort of security is not a Grid.

Secure data channels should always be optional, as they may be impractical in some circumstances. Should data channels always be authenticated, but not encrypted?

(Is this possible with TLS?)

May be important for getting through firewalls

Others?

3.9 Bindings

Client

Most of the Core Functions will be defined in terms of protocols and data structures, and this provides the basic uniformity required of Grids.

However, there will be many ways to use these Core Functions. For example

- Globus toolkit's C language [35]
- CoG kit's Java interface to the Globus functions [36]
- PyGlobus interface to the Globus functions [37]
- Arguably the OGSI work [38] represents a non-Globus interface to the Core functions

And there will be others.

Issues

It may be desirable, even necessary, to require that a minimal implementation of the core functions include the client side bindings for a representative set of programming styles. For example, the once given above: C, Java, Python, OGSI.

Resource

Is a C language implementation essential?

3.10 Other / Future Services as Core Grid Functions

These functions are noted for future discussion, but are not current practice in Grids.

Such functions might be generated by

- Classes of Grid resources that have not yet been considered
- New application types and/or uses that require new core functionality

Recall that criteria for core / minimal Grid functions are:

1. it be an important function
2. it cannot be built from existing Grid services, and therefore requires some operational elements on Grid resources (servers, libraries, etc.)

~~3.10.1Abstraction of Computing Resource Architecture~~

~~Provide for a mechanism to map a workflow onto different computing resource architectures.~~

~~Account, e.g., for how data is staged into and out of systems, how directory structures are set up for multiple tasks, how data is cached from one task to the next, etc.~~

~~UNICORE [6] addresses this issue.~~

~~3.10.2~~3.10.1 Transactional Messaging

Is transactional messaging a core Grid function?

~~3.10.3~~3.10.2 Reliable, Secure Multicast

Is reliable, secure multicast / secure group communication (see, e.g., [39]) a core Grid function?

~~3.10.4~~3.10.3 Checkpoint / Restart / Coordinated Recovery

Is checkpoint / restart / coordinated recovery a core Grid function?

~~3.10.5~~3.10.4 Structured Data Access

Are access methods for

- XML objects
- Time series

core Grid functions?

~~3.10.6~~3.10.5 Quality of Service

- Is this part of the service request?
- Part of scheduling?
- Separate?
- More than scheduling?

~~3.10.7~~3.10.6 Debug

Any special support needed for Grid debugging?

- Seems like local helper processes that are initiated by Grid job initiator will serve this purpose.
- Is there a permissions issue (how do I debug someone else's job?)

- Need to talk with Bob Hood (rhood@nas.nasa.gov) about this (he has built a Grid debugger)

~~3.10.8~~3.10.7 Communications channel "tapping"

- for debug, steering, analysis
- where is Nexus when we need it?

~~3.10.9~~3.10.8 Authorization

Issue: Is some type of access control a requirement? Is this a local control issue?

- Access control lists
- CAS [40]
- Attribute certificate based [41]

4. Security Considerations

Security is a fundamental aspect of this document, which describes the relationship of security to the core Grid functions.

5. Glossary

6. Author Contact Information

William E. Johnston
Lawrence Berkeley National Laboratory / NASA Ames Research Center
tel: +1-510-486-5014, fax: +1-603-719-1356
USMail: 1 Cyclotron Rd., MS 50B-2239, Berkeley, CA, 94720, USA
wejohnston@lbl.gov

John M. Brooke
Manchester Research Centre for Computational Science (MRCCS)
Manchester Computing, University of Manchester
Manchester M13 9PL, UK
Tel: +44 (0)161 275 6814 Fax: +44 (0)161 275 6800 Mobile 07765 220 227
http://www.csar.cfs.ac.uk/staff/brooke   Email: j.m.brooke@man.ac.uk

7. Acknowledgements

8. Notices

8.1 Intellectual Property Statement

8.2 Full Copyright Notice

9. Notes and References

[1] **Grid Information Services / MDS**, Globus Project. http://www.globus.org/mds/

Grid computing technologies enable wide-spread sharing and coordinated use of networked resources. Sharing relationships may be static and long-lived—e.g., among the major resource centers of a company or university—or highly dynamic: e.g., among the evolving membership of a scientific collaboration. In either case, the fact that users typically have little or no knowledge of the resources contributed by participants in the “virtual organization” (VO) poses a significant obstacle to their use. For this reason, information services designed to support the initial discovery and ongoing monitoring of the existence and characteristics of resources, services, computations, and other entities are a vital part of a Grid system. ("Grid Information Services for Distributed Resource Sharing" - http://www.globus.org/research/papers/MDS-HPDC.pdf)

The Monitoring and Discovery Service architecture addresses the unique requirements of Grid environments. Its architecture consists of two basic elements:

- A large, distributed collection of generic information providers provide access to information about individual entities, via local operations or gateways to other information sources (e.g., SNMP queries). Information is structured in term of a standard data model, taken from LDAP: an entity is described by a set of "objects" comprised of typed attribute-value pairs.

- Higher-level services, collect, manage, index, and/or respond to information provided by one or more information providers. We distinguish in particular aggregate directory services, which facilitate resource discovery and monitoring for VOs by implementing both generic and specialized views and search methods for a collection of resources. Other higher-level services can use this information and/or information obtained directly from providers for the purposes of brokering, monitoring, troubleshooting, etc.

Interactions between higher-level services (or users) and providers are defined in terms of two basic protocols: a soft-state registration protocol for identifying entities participating in the information service, and an enquiry protocol for retrieval of information about those entities, whether via query or subscription. In brief, a provider uses the registration protocol to notify higher-level services of its existence; a higher-level service uses the enquiry protocol to obtain information about the entities known to a provider, which it merges into its aggregate view. Integration with the Grid Security Infrastructure (GSI) provides for authentication and access control to information.

[2] **Super Scheduler Steps/Framework**, J. Schopf. http://www.mcs.anl.gov/~jms/ggf-sched/WD/schedwd.8.5.doc

http://www.mcs.anl.gov/~jms/ggf-sched/WD/schedwd.8.5.pdf

Overview of current user practices for scheduling across administrative domains. GGF document.

[3] **SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems**, K. Czajkowski, I. Foster, V. Sander, C. Kesselman and S. Tuecke. In *8th Workshop on Job Scheduling Strategies for Parallel Processing*. 2002. Edinburgh, Scotland.http://www.globus.org/research/papers/jsspp02-snap-preprint.pdf

A fundamental problem with distributed applications is to map activities such as computation or data transfer onto a set of resources that will meet the application's requirement for performance, cost, security, or other quality of service metrics. An application or client must engage in a multi-phase negotiation process with resource managers, as it discovers, reserves, acquires, configures, monitors, and potentially renegotiates resource access. Current approaches to resource management tend to specialize for specific classes of resource (processor, network, etc.), and have addressed coordination across resources in a limited fashion, if at all. We present a generalized resource management model in which resource interactions are mapped onto a well defined set of platform-independent service level agreements (SLAs). We instantiate this model in the Service Negotiation and Acquisition Protocol (SNAP) which provides lifetime management and an at-most-once creation semantics for remote SLAs. The result is a resource management framework for distributed systems that we believe is more powerful and general than current approaches. We explain how SNAP can be deployed within the context of the Globus Toolkit.

[4] **The Portable Batch Scheduler**. http://www.pbspro.com/tech_overview.html

The purpose of the PBS system is to provide additional controls over initiating or scheduling execution of batch jobs; and to allow routing of those jobs between different hosts [that run administratively coupled instances of PBS]. The batch system allows a site to define and implement policy as to what types of resources and how much of each resource can be used by different jobs. The batch system also provides a mechanism with which a user can insure a job will have access to the resources required to complete.

[5] **Maui Silver Metascheduler**. http://www.supercluster.org/documentation/silver/silveroverview.html

Silver is an advance reservation metascheduler. Its design allows it to load balance workload across multiple systems in completely independent administrative domains. How much or how little a system participates in this load sharing activity is completely up to the local administration. All workload is tracked and accounted for allowing 'allocation' exchanges to take place between the active sites.

[6] **UNICORE**. http://www.unicore.de/

UNICORE lets the user prepare or modify structured jobs through a graphical user interface on a local Unix workstation or a Windows PC. Jobs can be submitted to any of the platforms of a UNICORE GRID and the user can monitor and control the submitted jobs through the job monitor part of the client.

A UNICORE job contains a number of interdependent tasks. The dependencies indicate temporal relations or data transfer. Currently, execution of scripts, compile, link, execute tasks and data transfer directives are supported. An execution system request associated with a job specifies where its tasks are to be run. Tasks can be grouped into sub-jobs, creating a hierarchical job structure and allowing different steps to execute on different systems within the UNICORE GRID.

[7] **Policy Engine: a framework for authorization, accounting policy specification and evaluation in Grids**, B. Sundaram and B. M. Chapman. In *GRID2001, 2nd IEEE Workshop on Grid Computing*. 2001. Denver, Colorado, USA.http://www.cs.uh.edu/~ezgrid/PolicyEngine.pdf

We have developed a policy-based decision framework that provides authorization and cost-based accounting in the EZGrid system, a resource broker for metacompuintg. Primarily, this work allows the administrators and the owners to exercise more control over their resources by dictating usage permissions and/or restrictions on a grid environment. The mechanism is independent of the applications and the heterogeneous target domains. The EZGrid resource broker uses the policy engine to evaluate authorization policies of the remote sites in the process of making resource choices. Globus Access to Secondary Storage (GASS) is used as the backend for staging policy files, if needed, from the remote site to which authorization is required.

[8] **EZ-Grid Project: resource brokerage for multi-site computing**, EZ-Grid. http://www.cs.uh.edu/~ezgrid/arch.html

This project is to develop a smart resource broker built on existing Globus information and job management tools. Of particular relevance to a Grid Common Access service is the policy engine that maps site policies to users. See B. Sundaram, B. M. Chapman, Policy Engine: a framework for authorization, accounting policy specification and evaluation in Grids, in GRID2001, 2nd IEEE Workshop on Grid Computing, Denver 2001, Ed. Craig Lee.

[9] **Globus Resource Allocation Manager (GRAM)**, Globus Project. 2002. http://www-fp.globus.org/gram/overview.html

The Globus Resource Allocation Manager (GRAM) is the lowest level of Globus resource management architecture. GRAM allows you to run jobs remotely, providing an API for submitting, monitoring, and terminating your job.

To run a job remotely, a GRAM gatekeeper (server) must be running on a remote computer, listening at a port; and the application needs to be compiled on that remote machine. The execution begins when a GRAM user application runs on the local machine, sending a job request to the remote computer.

The request is sent to the gatekeeper of the remote computer. The gatekeeper handles the request and creates a job manager for the job. The job manager starts and monitors the remote program, communicating state changes back to the user on the local machine. When the remote application terminates, normally or by failing, the job manager terminates as well.

The executable, stdin and stdout, as well as the name and port of the remote computer, are specified as part of the job request. The job request is handled by the gatekeeper, which creates a job manager for the new job. The job manager handles the execution of the job, as well as any communication with the user.

[10] **A Network-Aware Distributed Storage Cache for Data Intensive Environments**, B. Tierney, J. Lee, B. Crowley, M. Holding, J. Hylton and F. Drake. In *Proc. 8th IEEE Symp. on High Performance Distributed Computing*. 1999.http://www-didc.lbl.gov/papers/dpss.hpdc99.pdf

[11] **The GridFTP Protocol and Software**, Globus Project. 2002. http://www.globus.org/datagrid/gridftp.html

GridFTP is a high-performance, secure, reliable data transfer protocol optimized for high-bandwidth wide-area networks. The GridFTP protocol is based on FTP, the highly-popular Internet file transfer protocol. We have selected a set of protocol features and extensions defined already in IETF RFCs and added a few additional features to meet requirement from current data grid projects.

[12] **The Storage Resource Broker**. http://www.npaci.edu/DICE/SRB/

The SDSC Storage Resource Broker (SRB) is a client-server middleware that provides a uniform interface for connecting to heterogeneous data resources over a network and accessing replicated data sets. SRB, in conjunction with the Metadata Catalog (MCAT), provides a way to access data sets and resources based on their attributes rather than their names or physical locations.

[13] **Grid Data Services - Relational Database Management**, B. Collins, A. Borley, N. Hardman, A. Knox, S. Laws, J. Magowan, M. Oevers and E. Zaluska. http://www.cs.man.ac.uk/grid-db/papers/grdb.pdf

This paper discusses issues associated with the development of relational database services, including usage scenarios.

[14] **Project Spitfite - Towards Grid Web Service Databases**, W. H. Bell, D. Bosio, W. Hoschek, P. Kunszt, G. McCance and M. Silander. 2002. http://www.cs.man.ac.uk/grid-db/papers/ggf5-spitfire.pdf

This paper describes the Spitfire grid database access service for relational databases.

This is an EU DataGrid project. See http://hep-proj-spitfire.web.cern.ch/hep-proj-spitfire/server/doc/

[15] **Pursuit of a Scalable High Performance Multi-Petabyte Database**, A. Hanushevsky and M. Nowak. In *Sixteenth IEEE Mass Storage Systems Symposium*. 1999.http://www.slac.stanford.edu/BFROOT/www/Public/Computing/Databases/proceedings/ieee_16mssc99.pdf

When the BaBar experiment at the Stanford Linear Accelerator Center starts in April 1999, it will generate approximately 200TB/year of data at a rate of 10MB/sec for 10 years. A mere six years later, CERN, the European Laboratory for Particle Physics, will start an experiment whose data storage requirements are two orders of magnitude larger. In both experiments, all of the data will reside in Objectivity databases accessible via the Advanced Multi-threaded Server (AMS). The quantity and rate at which the data is produced requires the use of a high performance hierarchical mass storage system in place of a standard Unix file system. Furthermore, the distributed nature of the experiment, involving scientists from 80 Institutions in 10 countries, also requires an extended security infrastructure not commonly found in standard Unix file systems. The combination of challenges that must be overcome in order to effectively deal with a multi-petabyte object oriented database is substantial. Our particular approach marries an optimized Unix file system with an industrial strength Mass Storage System. This paper describes what we had to do to create a robust and uniform system based on these components.

[16] **Creating Large Scale Database Servers**, J. Becla and A. Hanushevsky. In *Ninth IEEE International Symposium on High Performance Distributed Computing (HPDC'00)*. 2000.http://www.slac.stanford.edu/BFROOT/www/Public/Computing/Databases/proceedings/hpdc2000.pdf

The BaBar experiment at the Stanford Linear Accelerator Center (SLAC) is designed to perform a high precision investigation of the decays of the B-meson produced from electron-positron interactions. The experiment, started in May 1999, will generate approximately 300TB/year of data for 10 years. All of the data will reside in Objectivity databases accessible via the Advanced Multi-threaded Server (AMS). To date, over 70TB of data have been placed in Objectivity/DB, making it one of the largest databases in the world. Providing access to such a large quantity of data through a database server is a daunting task. A full-scale testbed environment had to be developed to tune various software parameters and a fundamental change had to occur in the AMS architecture to allow it to scale past several hundred terabytes of data. Additionally, several protocol extensions had to be implemented to provide practical access to large quantities of data. This paper will describe the design of the database, the changes that we needed to make in the AMS for scalability reasons, and how the lessons we learned would be applicable to virtually any kind of database server seeking to operate in the Petabyte region.

[17] **Objectivity Open File System**, A. Hanushevsky. In *HEPNT-HEPiX fall '99*. 1999.http://www-project.slac.stanford.edu/hepix/HEPiX99-oofs.ppt

Describes the architecture of a system that integrates Objectivity's OODB with HPSS.

[18] **Grid Monitoring Architecture Working Group**, Global Grid Forum. http://www-didc.lbl.gov/GGF-PERF/GMA-WG/

The Grid Monitoring Architecture working group is focused on producing a high-level architecture statement of the components and interfaces needed to promote interoperability between heterogeneous monitoring systems on the Grid. The main products of this work are the architecture document itself, and accompanying case studies that illustrate the concrete application of the architecture to monitoring problems.

[19] **Monitoring Data Archives for Grid Environments**, J. Lee, D. Gunter, M. Stoufer and B. Tierney. In *SC2002*. 2002.http://www-didc.lbl.gov/publications.html

Developers and users of high-performance distributed systems often observe performance problems such as unexpectedly low throughput or high latency. To determine the source of these performance problems, detailed end-to-end monitoring data from applications, networks, operating systems, and hardware must be correlated across time and space. Researchers need to be able to view and compare this very detailed monitoring data from a variety of angles. To solve this problem, we propose a relational monitoring data archive that is designed to efficiently handle high-volume streams of monitoring data. In this paper we present an instrumentation and event archive service that can be used to collect and aggregate detailed end-to-end monitoring information from distributed applications. This archive service is designed to be scalable and fault tolerant. We also show how the archive is based on "Grid Monitoring Architecture" defined by the Global Grid Forum.

[20] **Distributed Monitoring Framework (DMF)**, Lawrence Berkeley National Lab. http://www-didc.lbl.gov/DMF/

The goal of the Distributed Monitoring Framework is to improve end-to-end data throughput for data intensive applications in a high-speed WAN environments, and to provide the ability to do performance analysis and fault detection in a Grid computing environment. This monitoring framework will provide accurate, detailed, and adaptive monitoring of all of distributed computing components, including the network. Analysis tools will be able to use this monitoring data for real-time analysis, anomaly identification, and response.

Many of the components of the DMF have already been prototyped or implemented by the DIDC Group. The NetLogger Toolkit includes application sensors, some system and network sensors, a powerful event visualization tool, and a simple event archive. The Network characterization Service has proven to be a very useful hop-by-hop network sensor. Our work on the Global Grid Forum Grid Monitoring Architecture (GMA) addressed the event management system. JAMM (Java Agents for Monitoring Management) is preliminary work on sensor management. The Enable project produced a simple network tuning advice service.

[21] **Information and Monitoring Services Architecture**, European Union DataGrid - WP3. http://hepunx.rl.ac.uk/edg/wp3/documentation/doc/arch/index.html

The aim of this work package is to specify, develop, integrate and test tools and infrastructure to enable end-user and administrator access to status and error information in a Grid environment and to provide an environment in which application monitoring can be carried out. This will permit both job performance optimisation as well as allowing for problem tracing and is crucial to facilitating high performance Grid computing.

[22] **A Framework for Control and Observation in Distributed Environments**, W. Smith. NASA Ames Research Center. http://www.nas.nasa.gov/~wwsmith/papers.html

A GGF, GMA implementation.

[23] **DOE Science Grid PKI Certificate Policy And Certification Practice Statement**. http://www.doegrids.org/

This document represents the policy for the DOE Science Grid Certification Authority operated by ESnet. It addresses Certificate Policy (CP) and Certification Practice Statement (CPS). The CP is a named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements. For example, a particular certificate policy might indicate applicability of a type of certificate to the authentication of electronic data interchange transactions for the trading of goods within a given price range. The CPS is a statement of the practices, which a certification authority employs in issuing certificates.

[24] **ESnet's SciDAC PKI & Directory Project - Homepage**, T. Genovese and M. Helm. DOE Energy Sciences Network. http://envisage.es.net/

This is the ESnet PKI project site. ESnet is building a Public Key Infrastructure service to support the DOE Science Grid, SciDAC projects and other DOE research efforts. The main goal is to provide DOE scientist and engineers Identity and Service certificates that allow them to participate in the growing national and international computational Grids.

[25] **Certification Authorities**, European Union DataGrid. 2002. http://marianne.in2p3.fr/datagrid/ca/ca-table-ca.html

The current list of EU DataGrid recognized CAs and their certificates.

[26] **Certification Authorities Acceptance and Feature Matrices**, European Union DataGrid. 2002. http://www.cs.tcd.ie/coghlan/cps-matrix/

The Acceptance and Feature matrices are key aspects of establishing cross-site trust.

[27] **Grid Security Infrastructure (GSI)**, Globus Project. http://www.globus.org/security/

The primary elements of the GSI are identity certificates, mutual authentication, confidential communication, delegation, and single sign-on.

GSI is based on public key encryption, X.509 certificates, and the Secure Sockets Layer (SSL) communication protocol. Extensions to these standards have been added for single sign-on and delegation. The Globus Toolkit's implementation of the GSI adheres to the Generic Security Service API (GSS-API), which is a standard API for security systems promoted by the Internet Engineering Task Force (IETF).

[28] **OpenSSL**. http://www.openssl.org/

The OpenSSL Project is a collaborative effort to develop a robust, commercial-grade, full-featured, and Open Source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols as well as a full-strength general purpose cryptography library. The project is managed by a worldwide community of volunteers that use the Internet to communicate, plan, and develop the OpenSSL toolkit and its related documentation.

[29] **Internet X.509 Public Key Infrastructure Proxy Certificate Profile**, S. Tuecke, D. Engert, I. Foster, V. Welch, M. Thompson, L. Pearlman and C. Kesselman. February 2002. http://www.gridforum.org/security/ggf4_2002-02/draft-ietf-pkix-proxy-02.txt

http://www.gridforum.org/security/ggf4_2002-02/draft-ietf-pkix-proxy-02.pdf

A technical specification draft of the X.509 certificate extensions required to support proxies, which is used for GSI single sign-on and delegation.

[30] **GSS-API Extensions**, S. Meder, V. Welch, S. Tuecke and D. Engert. February 2002. http://www.gridforum.org/security/ggf4_2002-02/draft-ggf-gss-extensions-05.doc

http://www.gridforum.org/security/ggf4_2002-02/draft-ggf-gss-extensions-05.pdf

This document defines extensions to RFC 2743, Generic Security Service Application Program Interface Version 2, Update 1. Extensions include: credential export and import of credentials; delegation at any time; credential extensions (e.g. restrictions) handling.

[31] **GSI Online Credential Retrieval - Requirements**, J. Basney. February 2002. http://www.gridforum.org/security/ggf4_2002-02/draft-ggf-gsi-ocr-requirements-01.doc

http://www.gridforum.org/security/ggf4_2002-02/draft-ggf-gsi-ocr-requirements-01.pdf

An online credential retrieval (OCR) service gives users secure and convenient access to the credentials they need for authentication. To make credentials available, the service either stores the credentials in a secure repository or generates new credentials on request.

This memo defines requirements for online credential retrieval services that provide secure access to X.509 credentials in the Grid Security Infrastructure (GSI).

[32] **Generic Security Service Application Program Interface, Version 2**, J. Linn. http://www.ietf.org/rfc/rfc2078.txt?number=2078

The Generic Security Service Application Program Interface (GSS-API), as defined in RFC-1508, provides security services to callers in a generic fashion, supportable with a range of underlying mechanisms and technologies and hence allowing source-level portability of applications to different environments. This specification defines GSS-API services and primitives at a level independent of underlying mechanism and programming language environment, and is to be complemented by other, related specifications:

documents defining specific parameter bindings for particular language environments

documents defining token formats, protocols, and procedures to be implemented in order to realize GSS-API services atop particular security mechanisms

[33] PKI.

Public-Key certificate infrastructure ("PKI") provides the tools to create and manage digitally signed certificates. For identity authentication, a certification authority generates a certificate (most commonly an X.509 certificate) containing the name (usually X.500 distinguished name) of an entity (e.g. user) and that entity's public key. The CA then signs this "certificate" and publishes it (usually in an LDAP directory service). These are the basic components of PKI, and allow the entity to prove its identity, independent of location or system. For more information, see, e.g., RSA Lab's "Frequently Asked Questions About Today's Cryptography" http://www.rsa.com/rsalabs/faq/, Computer Communications Security: Principles, Standards, Protocols, and Techniques. W. Ford, Prentice-Hall, Englewood Cliffs, New Jersey, 07632, 1995, or Applied Cryptography, B. Schneier, John Wiley & Sons, 1996.

[34] **GSI-Enabled OpenSSH**, NCSA. http://www.ncsa.uiuc.edu/Divisions/ACES/GSI/openssh/

NCSA maintains a patch to OpenSSH that adds support for GSI authentication.

[35] **The Globus Project**, Globus. http://www.globus.org

The Globus project is developing fundamental technologies needed to build computational grids. Grids are persistent environments that enable software applications to integrate instruments, displays, computational and information resources that are managed by diverse organizations in widespread locations.

[36] **A Java Commodity Grid Kit**, G. v. Laszewski, I. Foster, J. Gawor and P. Lane. Concurrency: Experience and Practice, 2001. http://www.globus.org/cog/documentation/papers/index.html

[37] **Python Globus (pyGlobus)**, K. Jackson. Lawrence Berkeley National Laboratory. http://www-itg.lbl.gov/gtg/projects/pyGlobus/index.html

- Provide a clean object-oriented interface to the Globus toolkit.

- Provide similar performance to using the underlying C code as much as possible.

- Minimize the number of changes necessary when aspects of Globus change.

- Where possible, make Globus as natural to use from Python as possible.

- For example, the gassFile module allows the manipulation of remote GASS files as Python file objects.

[38] **Open Grid Service Interface Working Group**, Global Grid Forum. http://www.gridforum.org/ogsi-wg/

The purpose of the OGSI Working Group is to review and refine the Grid Service Specification and other documents that derive from this specification, including OGSA-infrastructure-related technical specifications and supporting informational documents.

[39] **Reliable and Secure Group Communication**, D. Agarwal, K. Berket and O. Chevassut. http://www-itg.lbl.gov/CIF/GroupComm

[40] **Community Authorization Service (CAS)**, Globus Project. 2002. http://www.globus.org/security/CAS/

CAS allows resource providers to specify course-grained access control policies in terms of communities as a whole, delegating fine-grained access control policy management to the community itself. Resource providers maintain ultimate authority over their resources but are spared day-to-day policy administration tasks (e.g. adding and deleting users, modifying user privileges). Briefly, the process is: 1) A CAS server is initiated for a community: a community representative acquires a GSI credential to represent that community as a whole, and then runs a CAS server using that community identity. 2) Resource providers grant privileges to the community. Each resource provider verifies that the holder of the community credential represents that community and that the community's policies are compatible with the resource provider's own policies. Once a trust relationship has been established, the resource provider then grants rights to the community identity, using normal local mechanisms (e.g. gridmap files and disk quotas, filesystem permissions, etc.). 3) Community representatives use the CAS to manage the community's trust relationships (e.g., to enroll users and resource providers into the community according to the community's standards) and grant fine-grained access control to resources. The CAS server is also used to manage its own access control policies; for example, community members who have the appropriate privileges may authorize additional community members to manage groups, grant permissions on some or all of the community's resources, etc. 4) When a user wants to access resources served by the CAS, that user makes a request to the CAS server. If the CAS server's database indicates that the user has the appropriate privileges, the CAS issues the user a GSI restricted proxy credential with an embedded policy giving the user the right to perform the requested actions. 5) The user then uses the credentials from the CAS to connect to the resource with any normal Globus tool (e.g. GridFTP). The resource then applies its local policy to determine the amount of access granted to the community, and further restricts that access based on the policy in the CAS credentials, This serves to limit the user's privileges to the intersection of those granted by the CAS to the user and those granted by the resource provider to the community.

[41] **Certificate-based Access Control for Widely Distributed Resources**, M. Thompson, W. Johnston, S. Mudumbai, G. Hoo, K. Jackson and A. Essiari. In *Eighth Usenix Security Symposium*. 1999.http://www-itg.lbl.gov/Akenti/papers.html